CREATORCON™
April 23-24, 2015 • Mandalay Bay • Las Vegas, NV

Lab Guide

Advanced Scripting

Steven Bell & Mark Amann

Lab instance: https://LabPrefix###.lab.service-now.com

Default Login / Password:

admin / Knowledge15

itil / Knowledge15

employee / Knowledge15

© 2015 ServiceNow, Inc. All rights reserved.

This

Page

Intentionally

Left

Blank

© 2015 ServiceNow, Inc. All rights reserved.

Requirements

The client would like to have two fields added to the Incident form. This would allow the user to enter two numbers to be compared. The comparison will be initiated by clicking a button located at the top of the form. The two fields will be labeled: Thing1 and Thing2. These will be placed after the Assigned To field. The button will be labeled: Thing Compare. The comparison is to be done on the server and is not to interfere with the user experience. The results are to be displayed beneath the Thing1 field. If there is an error that is to be placed beneath the Thing2 field. Next another button, labeled Display Ten Incidents, will be added to the form that will retrieve a list of the latest 10 incidents that are in a state of New or Active and have an Impact of High. These will be displayed in a new form.

Note: Normally the requirements would be numbered in order to show the traceability to the design, and the code (and ultimately to test). This has not been done here due to the brevity of the session.

Design

1. Create new update set to contain the changes, and allow the project to be moved to Production when testing has been completed.
2. Add two new fields to the Incident form after the Assigned To field. These will be labeled Thing1 and Thing2 respectively.
3. Add a new UI Action with the label: Thing Compare. This will fire an Ajax call to a Script Include, pass Thing1 and Thing2, do the comparison, and pass the results back. The results will be displayed beneath Thing1. An error result will be placed beneath Thing2.
4. Add a new UI Action with the label: Display Ten Incidents. This will fire an Ajax call to the same Script Include, and retrieve a list of the latest 10 incidents that have a state of 1 or 2 (New or Active), and have an Impact of 1 (High). Display the results to the user in a new form.
5. Create a Script Include that will have the following methods (functions):
    a. Receive two values via Ajax, compare the values, and pass back the results using JSON.
    b. Retrieve the latest 10 incident records that have the state of 1 or 2, and an impact of 1.

© 2015 ServiceNow, Inc. All rights reserved.

Lab Goal

The purpose of this lab is to show a best practice: Whenever you go to make any modifications to the ServiceNow application you should always create an update set prior to doing so. This helps you keep track of what has been modified, and will assist in promoting the modifications to another instance.

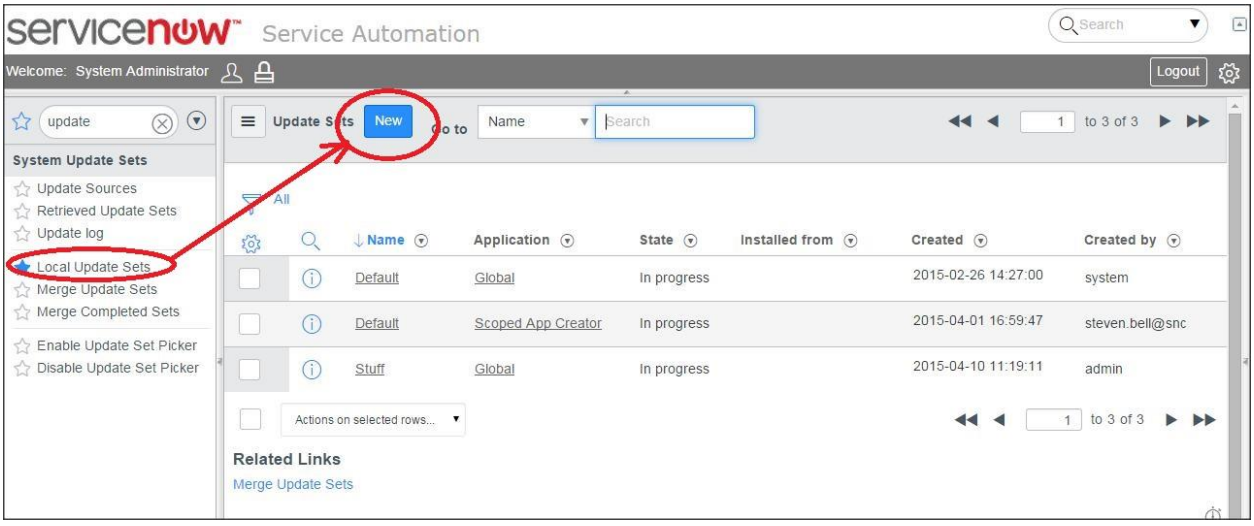
Lab 1.1 Create a New Update Set (Design #1)

1. Log onto your training instance.

2. Type the following in the navigation search: **update**. The System Update Sets application will be displayed in the navigation menu.

3. Click the following module link: **Local Update Sets**. A list view of the local update sets will be displayed. A best practice is to drag this link over to the Edge Bar for easier later reference.

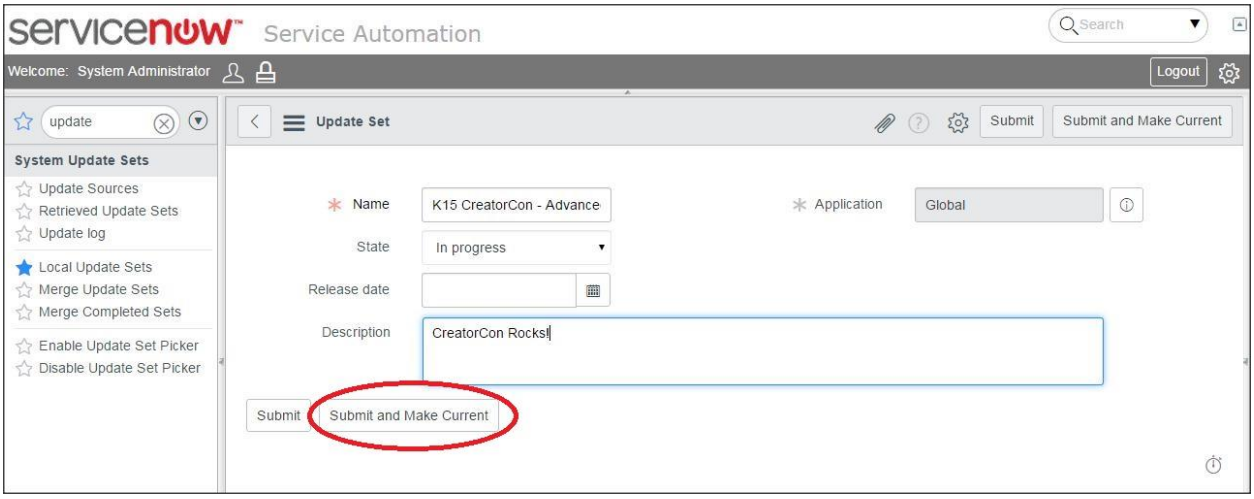
4. At the top of the List View click the **New** button. The create new Update Set form will be displayed.



© 2015 ServiceNow, Inc. All rights reserved.

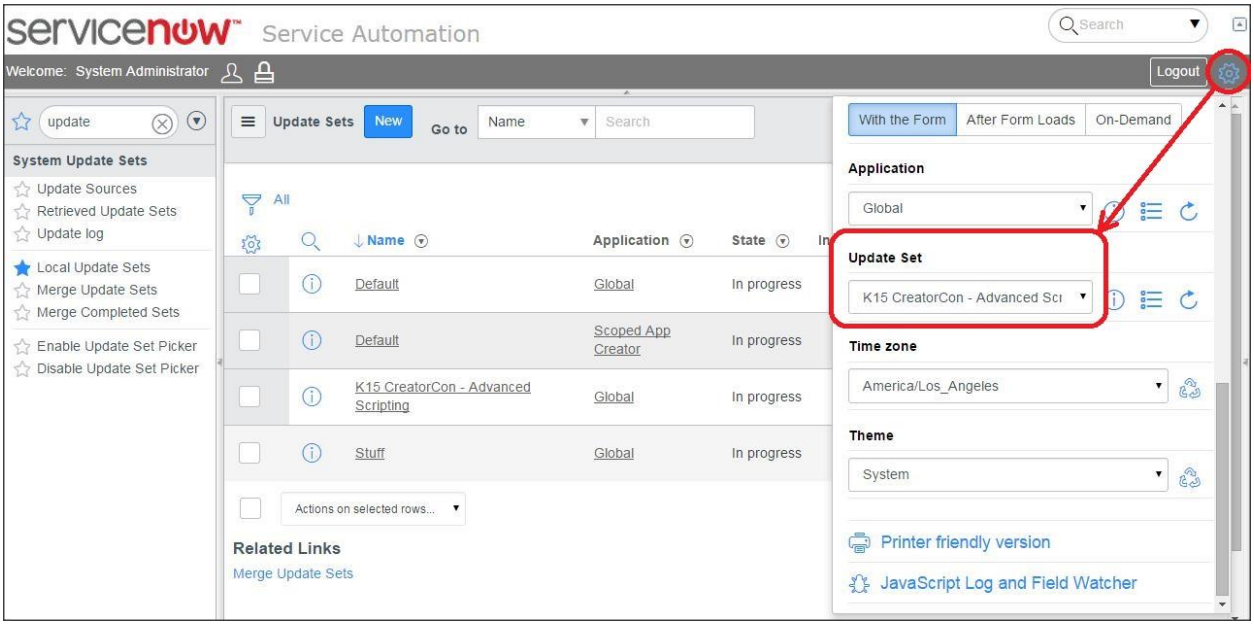
5. Fill in the following fields:
    - **Name**: Advanced Scripting Lab - <<your initials>> (or anything that you like)
    - **Description**: Advanced Scripting Lab results (again anything you like)

6. Click the **Submit And Make Current** button at the bottom of the form. This will create your new update set and make it the current one. You can verify this by clicking on the **Gear** button located in the upper left of your browser, and observing the current update set value.



7. You are done creating your update set!

8. Now one last step. You need to verify that the update set really is your current. Click on the **Gear** button in the upper right side. This opens the SeviceNow Configuration context menu. Scroll down and verify the current update set. It should look something like this:



© 2015 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 1.2 Script Include Basics (Design #5.a, #2, #3)

The purpose of this lab is to demonstrate:

- The creation of a Script Include
- Script Include Methods
- Byref / Byval function parameter passing
- JSON Encode/Decode
- Simple Object Passing via Ajax

Design Recap

Add two new fields to the Incident form after the Assigned To field. These will be labeled Thing1 and Thing2 respectively.

Add a new UI Action with the label: Thing Compare. This will fire an Ajax call to a Script Include, pass Thing1 and Thing2, do the comparison, and pass the results back. The results will be displayed beneath Thing1. An error result will be placed beneath Thing2.

Create a Script Include that will have the following methods (functions): Receive two values via Ajax, compare the values, and pass back the results using JSON.

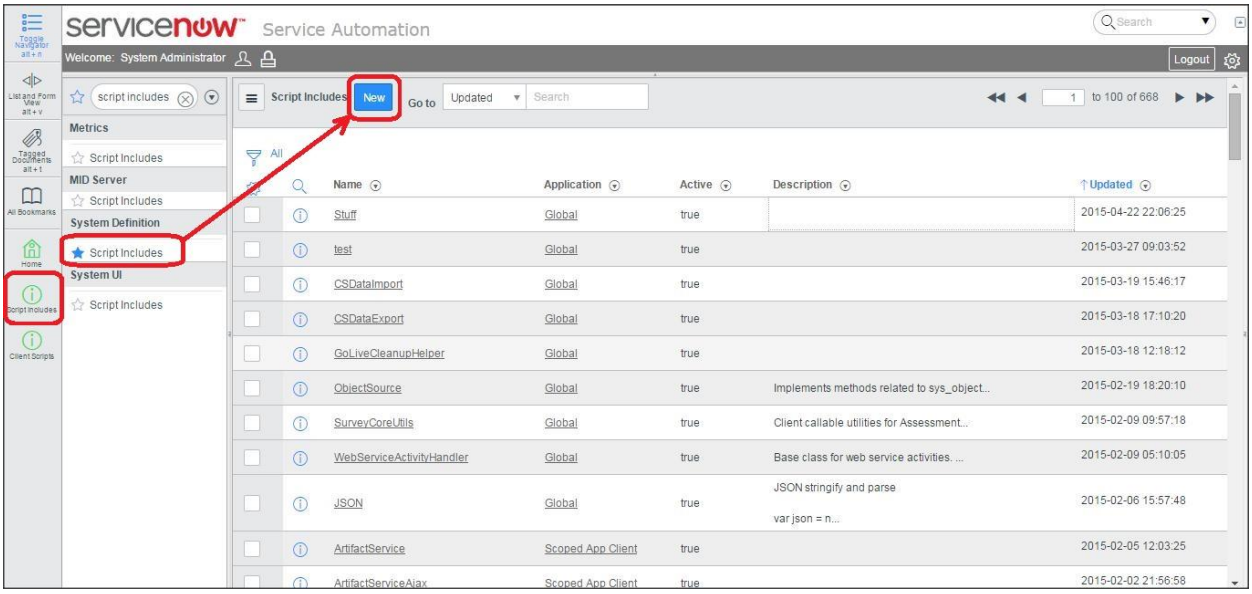
Create the Script Include

1. Type the following in the navigation search: **script includes**.

2. Under the System Definition module click the following module link: Script Includes. This will display a list view of the script includes table. A best practice is to drag this link over to the Edge Bar for easier later reference.

© 2015 ServiceNow, Inc. All rights reserved.

3. At the top of the List View click the **New** button. The create new Script Include form will be displayed.



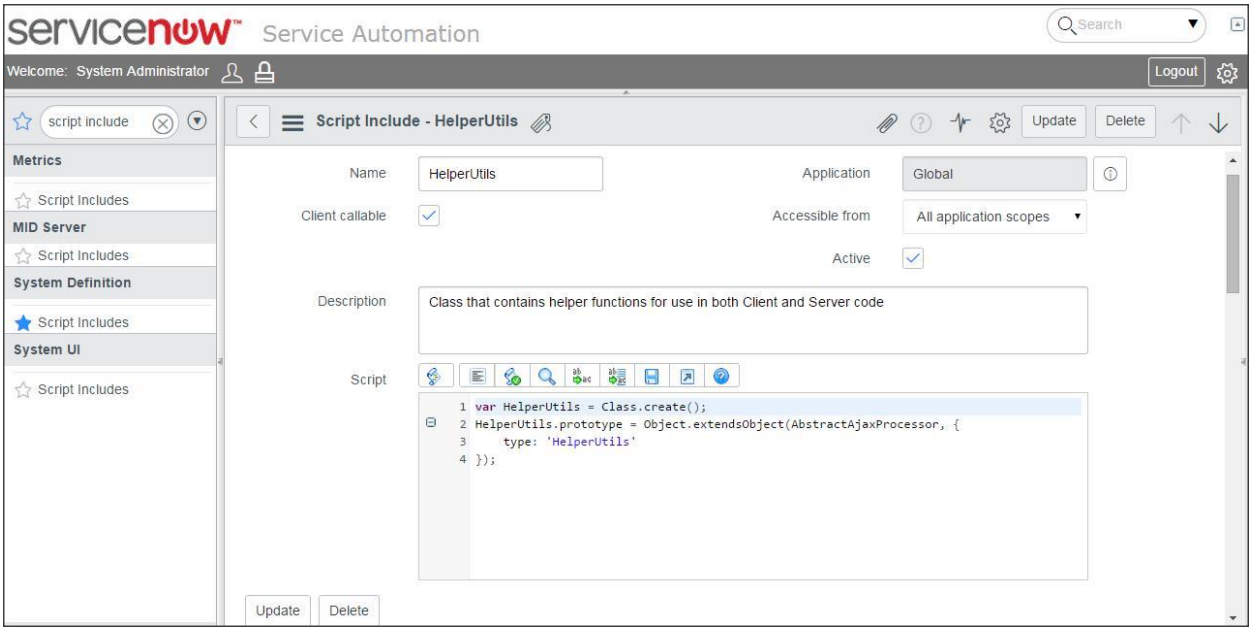
4. Fill in the following fields:
    - **Name**: HelperUtils. Note that when you leave the Name field that a template of a basic script include is placed into the Script field below, and that it has the HelperUtils name embedded throughout the code.
    - **Description**: Class that contains helper functions for use in both Client and Server code.
    - **Check**: Client callable
    - **Script**: Place a semi-colon at the end of line 7. The template doesn't do this and it causes a syntax error in the next check.
5. Click the **Check Syntax** button  to verify that the script syntax is correct. We are doing this as a good practice whenever the code is changed. The following dialog should be displayed:



6. Click okay to close the dialog.

© 2015 ServiceNow, Inc. All rights reserved.

7. Click the **Save** button on the form to save your work (or save from the form context menu).



8. You have now created your base Script Include to be used throughout the lab.

9. Now add the following code between lines 3 and 4 in the script field. If you don't want to type it in then ask your instructor for the url to download the code.

```
// internal helper functions
compareValues : function() {
    // get the AJAX passed in values from the client script
    var compareNum1 = this.getParameter('sys_param_value1') + '';
    var compareNum2 = this.getParameter('sys_param_value2') + '';

    // initialize the object
    var status = {message:'', error:''};

    // pass in the object (by reference), and the two strings (by value)
    this._compareValues(compareNum1, compareNum2, status);

    // call a function to do the JSON.encode
    return this._encode(status);
},

// example usage of ternary operator
_compareValues : function(value1, value2, status) {
    try {
          status.error += isNaN(parseInt(value1)) ? 'Value1 is a string! ' :
'Value1 is a number! ';
          status.error += isNaN(parseInt(value2)) ? 'Value2 is a string! ' :
'Value2 is a number! ';
          status.message += 'Value1 is ' + (value1 === value2 ? '' : 'not') + '
```

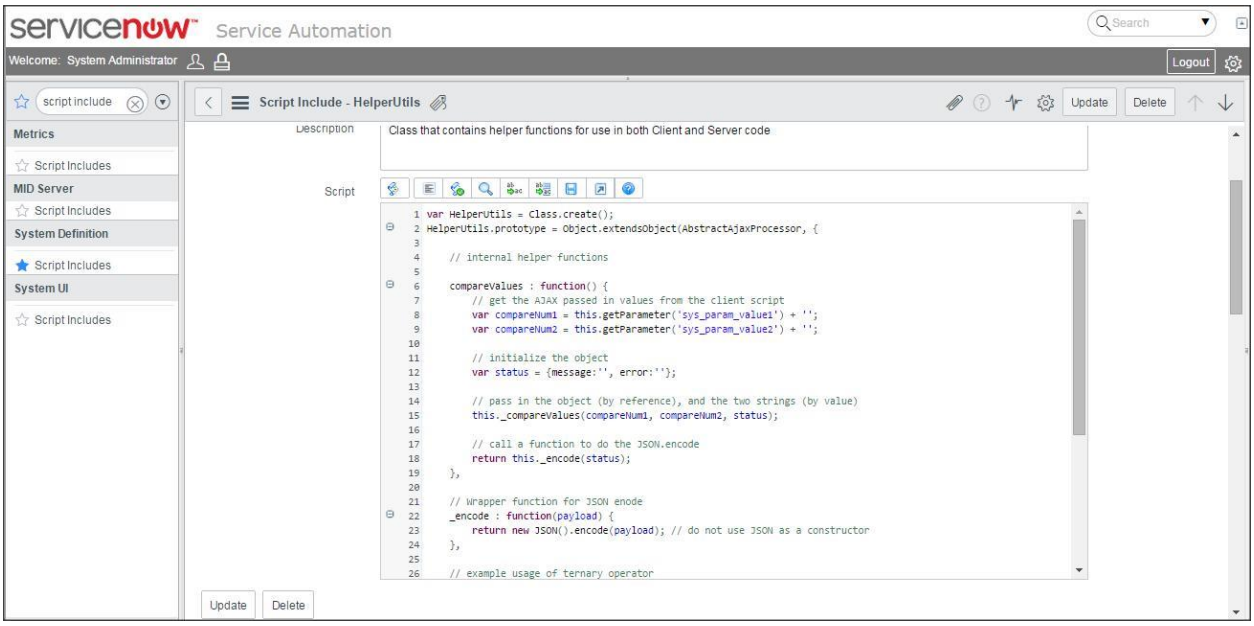
© 2015 ServiceNow, Inc. All rights reserved.

```
the same as Value2';
    }
    catch (err) {
          status.error = err;
          // scoped logging
          gs.error('---> SI HelperUtils._compareValues. ERROR: {0}', err);
    }
},

// Wrapper function for JSON enode
_encode : function(payload) {
    return new JSON().encode(payload); // do not use JSON as a constructor
},
```

WARNING: Make sure you get that comma on the end of the _encode method!

Your Script Include should look like this:



10. Syntax check your code then click the **Update** button to save your code.

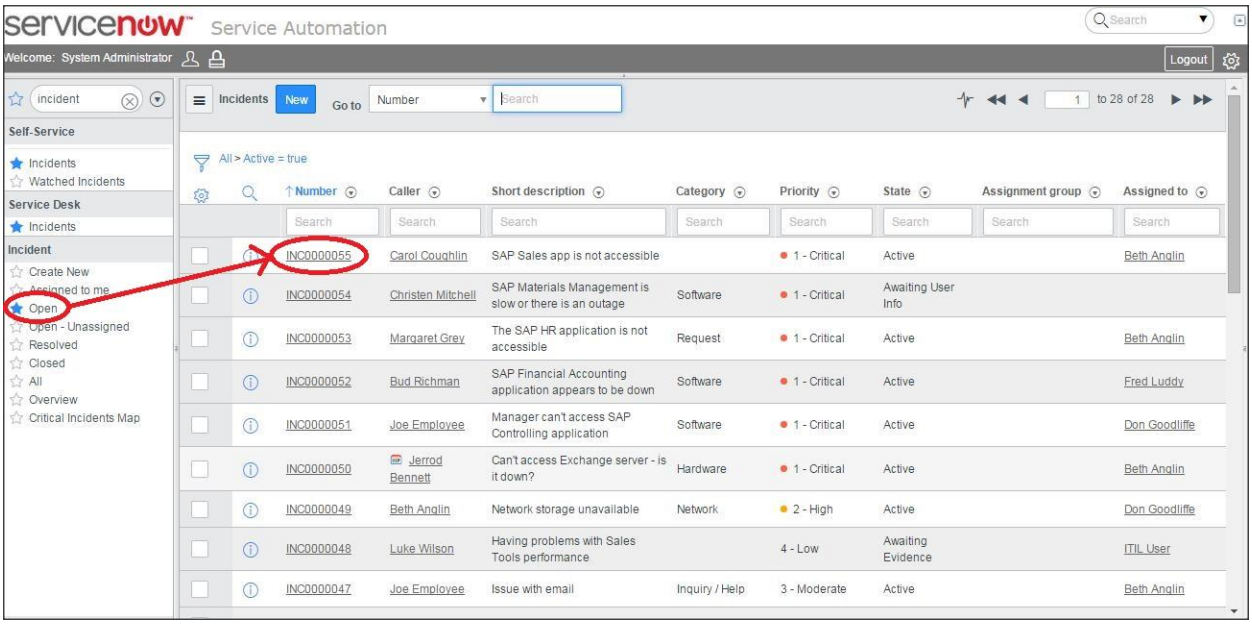
© 2015 ServiceNow, Inc. All rights reserved.

Add Two Fields to The Incident Form and Create a new Incident UI Action to Compare Them (Design #2, #3)

1. Type the following in the navigation search: **Incident**.

2. Navigate to **Incident > Open** (the greater-than sign signifies “then” and implies clicking on the given link). This will display a list of incidents.

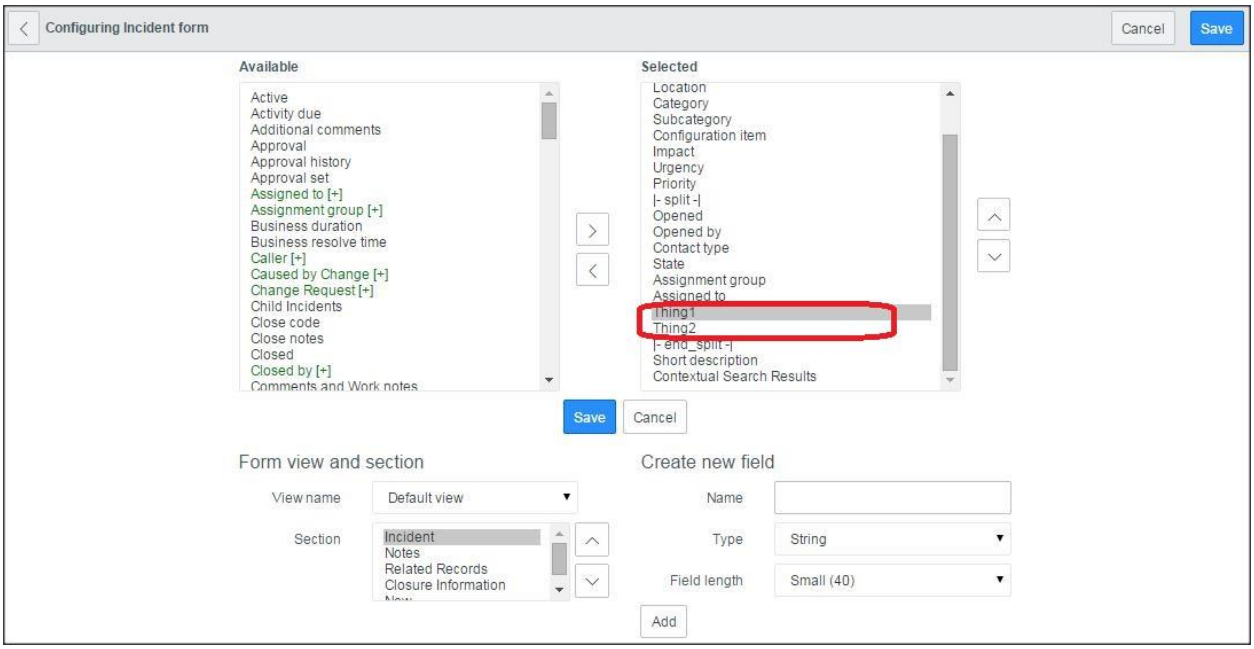
3. Click on the incident number of the first record in the list. This will display that incident in a form view.



4. Right-Click on the form banner to bring up the form context-menu. Choose: **Configure > Form Layout**. The Form Layout screen will be displayed.

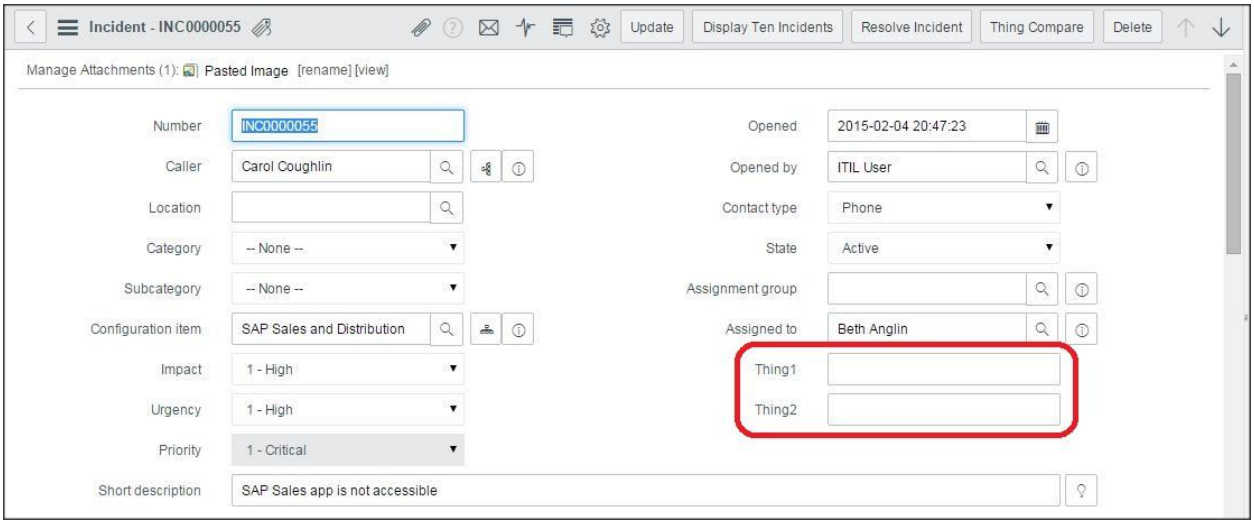
5. Add two new fields labeled **Thing1** and **Thing2**. **Type**: String. Place these after the Assigned To field.

© 2015 ServiceNow, Inc. All rights reserved.



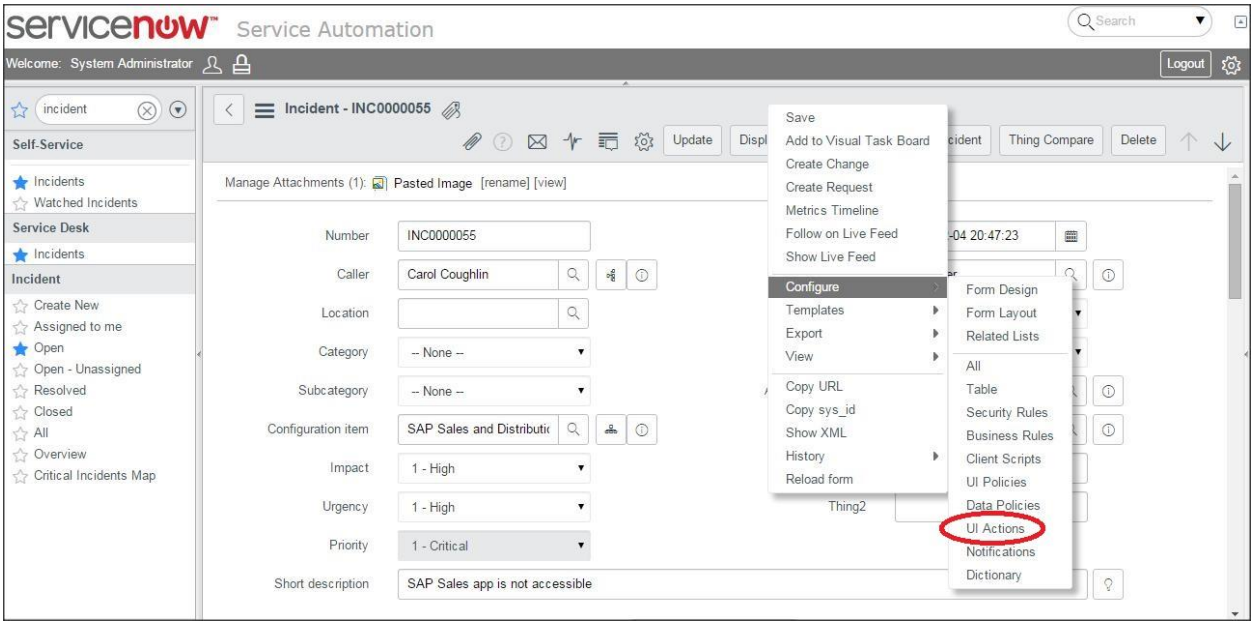
6. Click the **Save** button. This will save the changes and return you to the Incident form.

7. You should now see the two new fields: Thing1 and Thing2 placed after the Assigned To field on the form.



8. Right-Click on the form banner to bring up the form context-menu. Choose: **Configure > UI Actions**. A list of existing UI Actions will be displayed.

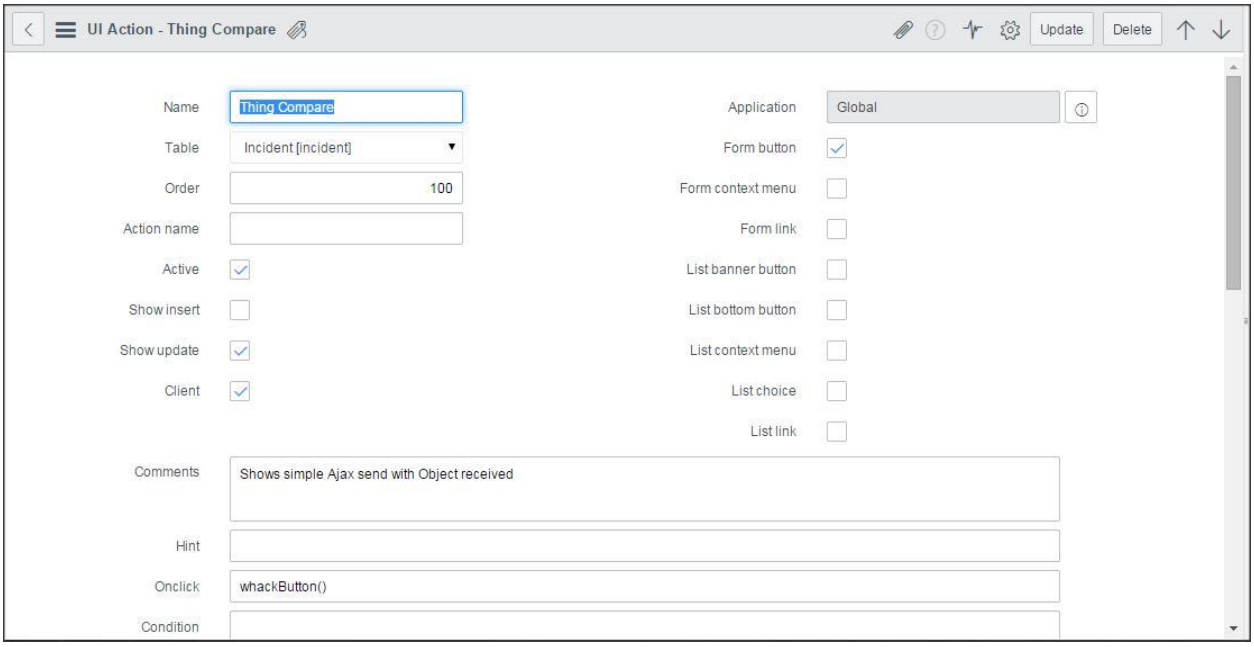
© 2015 ServiceNow, Inc. All rights reserved.



9. Click the **New** button at the top of the list view. This will open the new UI Action form.

10. Fill in the following fields:
    - **Name**: Thing Compare
    - **Table**: Incident (should be autofilled)
    - **Active**: checked
    - **Show Update**: checked
    - **Client**: checked
    - **Form button**: checked
    - **Comments**: Shows simple Ajax send with Object received
    - **Onclick**: whackButton()

© 2015 ServiceNow, Inc. All rights reserved.



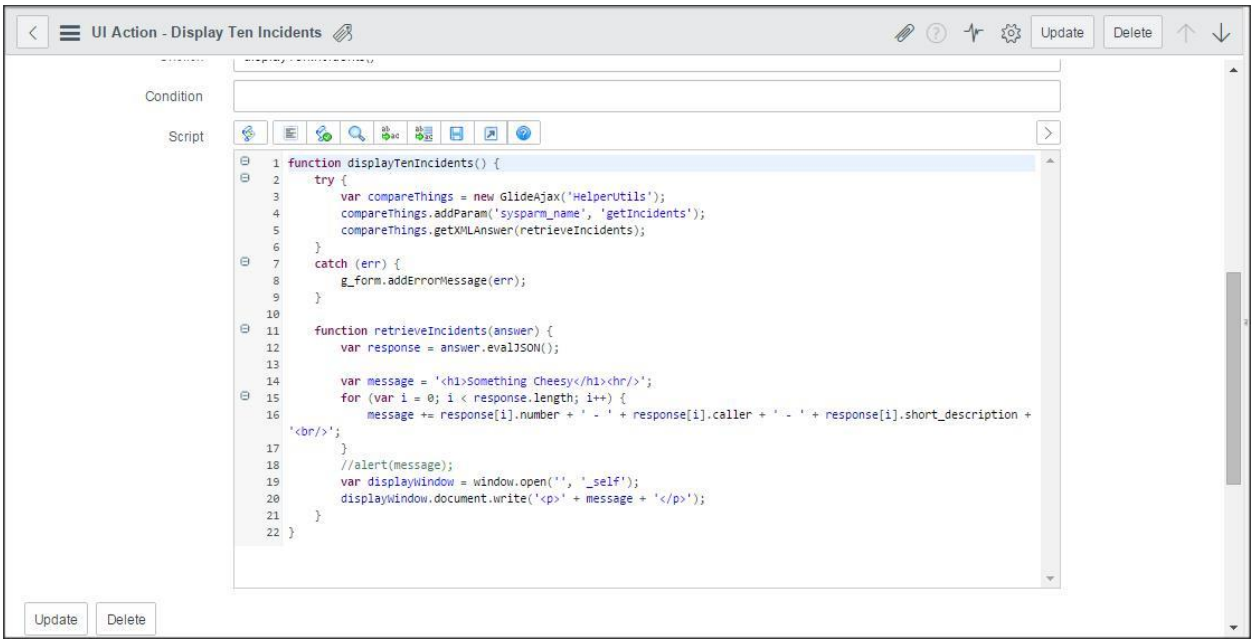
- **Script**:

```
function whackButton() {
    try {
          var compareThings = new GlideAjax('HelperUtils');
          compareThings.addParam('sysparm_name', 'compareValues');
          compareThings.addParam('sys_param_value1',
g_form.getValue('u_thing1'));
          compareThings.addParam('sys_param_value2',
g_form.getValue('u_thing2'));
          // Asynch call
          compareThings.getXMLAnswer(retrieveComparison);
    }
    catch (err) {
          g_form.addErrorMessage(err);
    }

    // callback function
    function retrieveComparison(answer) {
          var response = answer.evalJSON();
          var message = response.message + '';
          var error = response.error + '';

          g_form.hideFieldMsg('u_thing1');
          if (message != '') {
                g_form.showFieldMsg('u_thing1', message, 'info');
          }
          if (error != '') {
                g_form.showFieldMsg('u_thing2', error, 'error');
          }
    }
}
```

© 2015 ServiceNow, Inc. All rights reserved.

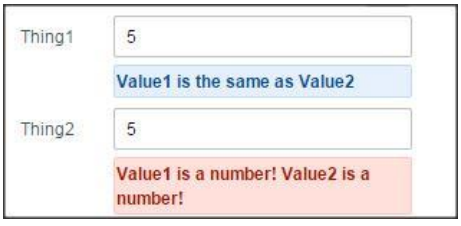


11. Do a syntax check on the code, then click the **Save** button, or right-click on the header and pick save from the context menu.

12. Navigate to Incidents and choose the first record again. You should now see a button at the top of the form labeled: Thing Compare.

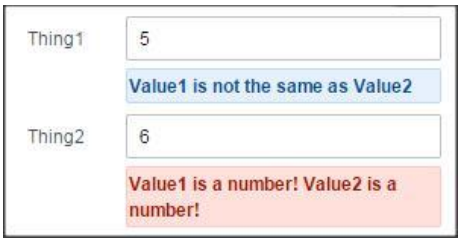
Testing

1. Fill in a number in Thing1 and the same number in Thing2 then click on the Thing Compare button. The message: will appear below Thing1.

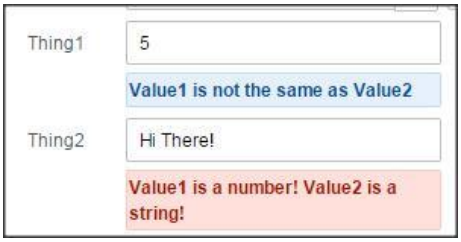


2. Fill in a number in Thing1 and a different number in Thing2 then click on the Thing Compare button. The message: will appear below Thing2.

© 2015 ServiceNow, Inc. All rights reserved.



3. Fill in a number in Thing1 and a string in Thing2 then click on the Thing Compare button. The message: will appear below Thing2.



© 2015 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 1.3
Retrieve an Object Array
(Design #5.b, #4)

The purpose of this lab is to demonstrate:

- Converting a GlideRecord object into an Object Array
- Passing an Object Array back via JSON
- Object reconstitution via evalJSON
- Displaying results
- Displaying error and message handling

Design Recap

4. Add a new UI Action with the label: Display Ten Incidents. This will fire an Ajax call to the same Script Include, and retrieve a list of the latest 10 incidents that have a state of 1 or 2 (New or Active), and have an Impact of 1 (High). Display the results to the user in a new form.

5. Create a Script Include that will have the following methods (functions):
    b. Retrieve the latest 10 incident records that have the state of 1 or 2, and an impact of 1.

Modify the Existing Script Include

1. Click on the **Script Includes** link on your Edge bar. This will bring up a list view of the Script Includes. Sort by Updated descending (most recent displayed first). Your HelperUtils Script Include should be displayed first.

2. Click on the **HelperUtils** Script Include to open it.

3. In the **Script** field add the following function after the **compareValues** function:

© 2015 ServiceNow, Inc. All rights reserved.

```
// example of creating an object array from a GlideRecord
getIncidents : function() {
    var incidentList = [];

    var incidents = new GlideRecord('incident');
    // demonstrates chaining, and the "IN" comparison
    incidents.addQuery('state', 'IN', '1, 2').addOrCondition('impact', '1');
    incidents.addActiveQuery();  // demonstrates the addActiveQuery function
    incidents.setLimit(10);  // demonstrates the setLimit function
    incidents.query();

    while (incidents.next()) {
          var incident = {};
          incident.number = incidents.getValue('number');
          incident.caller = JSUtil.notNil(incidents.caller_id) ?
incidents.caller_id.getDisplayValue() : 'no caller!';  // gets the name
          incident.short_description = incidents.getValue('short_description');
          incidentList.push(incident);
    }

    return this._encode(incidentList);
},
```

4. Syntax check your code then save it down.

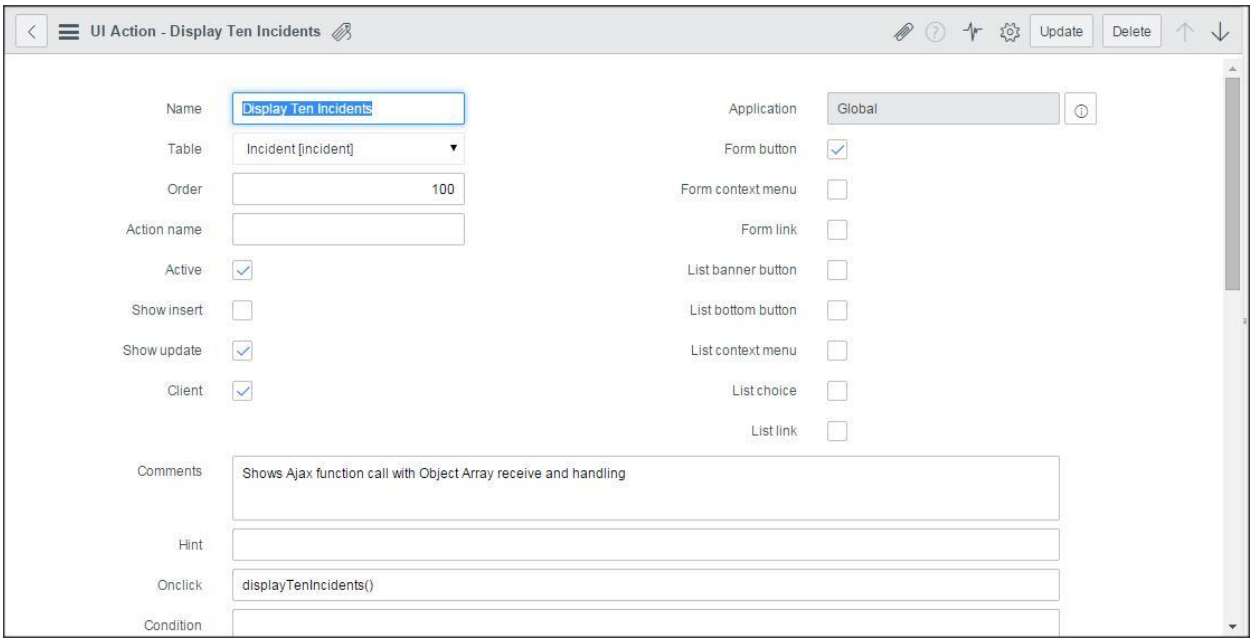
5. Now we will show a different way to add a UI Action to a form. In the navigation text box type: **UI Action**.

6. Navigate to **System Definition > UI Actions**. This will display a list view of all UI Actions. Sort by Updated descending. Note that your new UI Action Thing Compare shows at the top of the list.

7. Click on the **New** button to bring up an empty UI Action form.

8. Fill in the following fields:
    - **Name**: Display Ten Incidents
    - **Table**: Incident (will not be filled in initially)
    - **Active**: checked
    - **Show Update**: checked
    - **Client**: checked
    - **Form button**: checked
    - **Comments**: Shows Ajax function call with Object Array receive and handling
    - **Onclick**: displayTenIncidents()

© 2015 ServiceNow, Inc. All rights reserved.



- **Script**:

```
function displayTenIncidents() {
    try {
          var compareThings = new GlideAjax('HelperUtils');
          compareThings.addParam('sysparm_name', 'getIncidents');
          compareThings.getXMLAnswer(retrieveIncidents);
    }
    catch (err) {
          g_form.addErrorMessage(err);
    }

    function retrieveIncidents(answer) {
          var response = answer.evalJSON();

          var message = '<h1>Something Cheesy</h1><hr/>';
          for (var i = 0; i < response.length; i++) {
                message += response[i].number + ' - ' + response[i].caller + ' -
' + response[i].short_description + '<br/>';
          }
          //alert(message);
          var displayWindow = window.open('', '_self');
          displayWindow.document.write('<p>' + message + '</p>');
    }
}
```

© 2015 ServiceNow, Inc. All rights reserved.

Script

```
function displayTenIncidents() {
    try {
        var compareThings = new GlideAjax('HelperUtils');
        compareThings.addParam('sysparm_name', 'getIncident');
        compareThings.getXMLAnswer(retrieveIncidents);
    }
    catch (err) {
        g_form.addErrorMessage(err);
    }

    function retrieveIncidents(answer) {
        var response = answer.evalJSON();

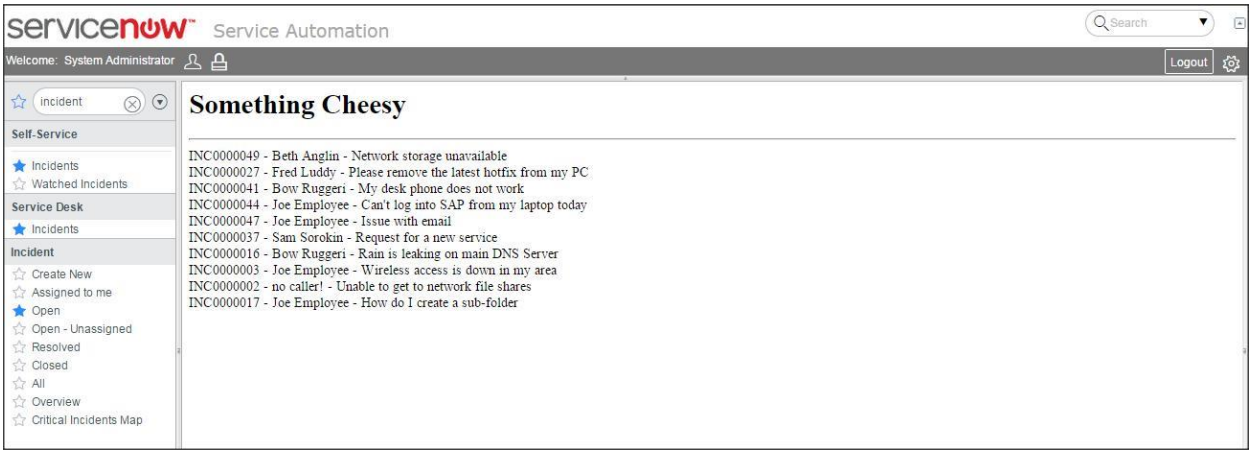
        var message = '<h1>Something Cheesy</h1><hr/>';
        for (var i = 0; i < response.length; i++) {
            message += response[i].number + ' - ' + response[i].caller + ' - ' + response[i].short_description +
'<br/>';
        }
        //alert(message);
        var displayWindow = window.open('', '_self');
        displayWindow.document.write('<p>' + message + '</p>');
    }
}
```

9. Do a syntax check on the code, then click the **Save** button, or right-click on the header and pick save from the context menu.

10. Navigate to Incidents and choose the first record again. You should now see a button at the top of the form labeled: Display Ten Incidents.

Testing

1. Click on the Display Ten Incidents button. A form will appear with a list of ten incidents ordered by most recent descending will be displayed.

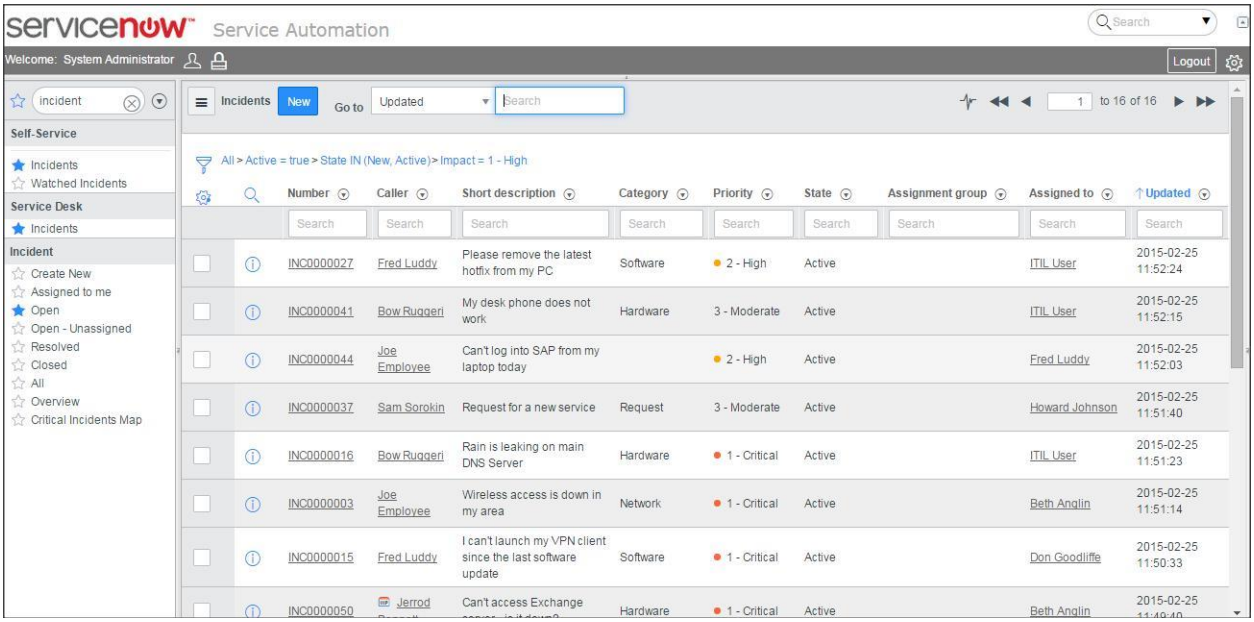


2. Navigate to the Incident open list view and use the following query to verify your results:

© 2015 ServiceNow, Inc. All rights reserved.

Active = true > State IN (New, Active) > Impact = 1 – High

The results should look like this (order by updated descending):



© 2015 ServiceNow, Inc. All rights reserved.

Lab Goal

The purpose of this lab is to demonstrate:

- The creation of a Script Include
- Script Include Methods
- Byref / Byval function parameter passing
- JSON Encode/Decode
- Simple Object Passing via Ajax

Lab 1.4
Cluster Information
[Optional]

Cluster Version, Patch, and Date Information

Sometimes it is necessary to really dig to find the information you are wanting to display for the user. In this lab you will investigate where certain system information is stored, how to interpret what you have found, and then code to retrieve the data.

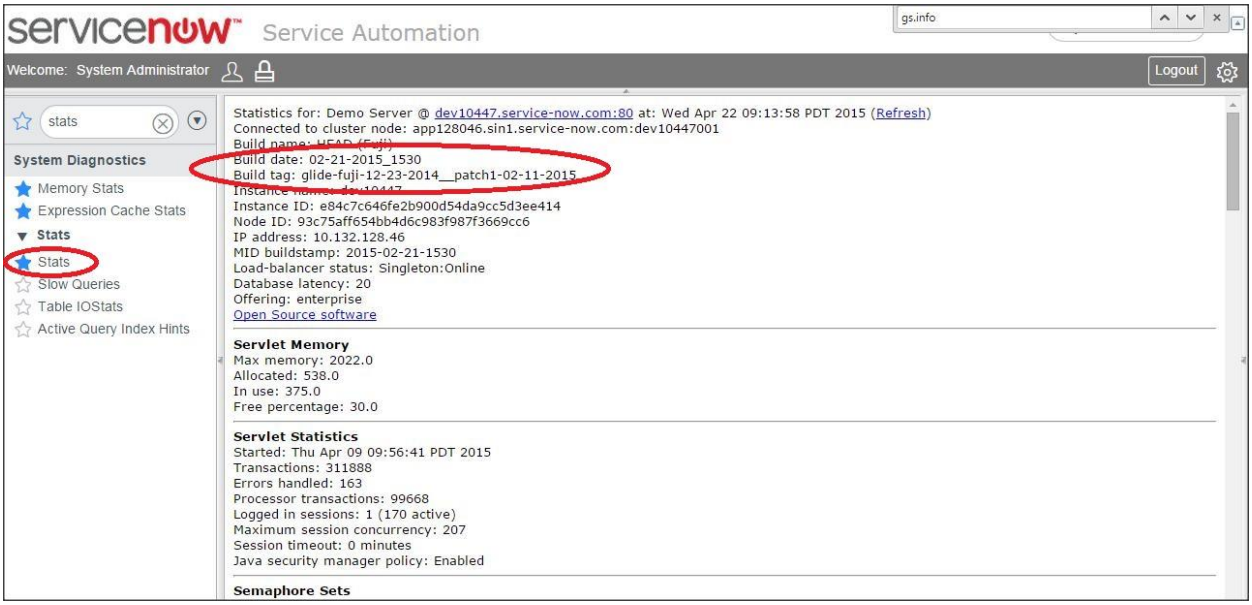
As an Admin it is possible to retrieve system information via the stats.do page. This page is not to be found in the UI Pages or UI Macros list views. So it appears to be built-in. But where is the information found on this page actually stored?

First let's go take a look at the page in question.

1. Type stats in the Nav Filter

2. In the navigator click on the Stats > Stats link.

© 2015 ServiceNow, Inc. All rights reserved.



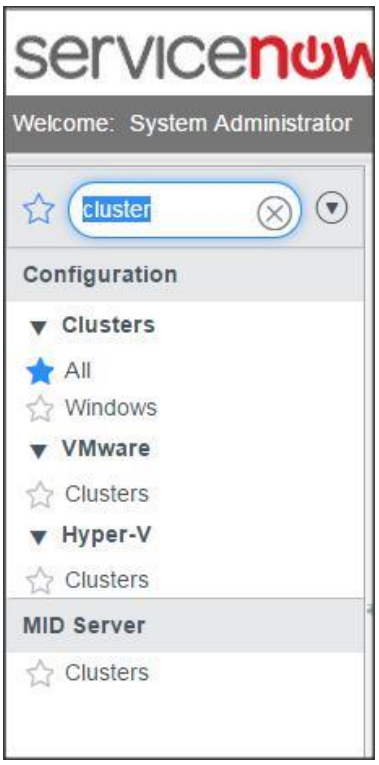
You will notice at the top that there is a lot of useful information here. Build name, build patch, build date, url, and a lot more. This is the "cluster" information. There are typically two servers clustered together for each ServiceNow instance. This allows for fail-over in case something should happen to the "active" node of the cluster.

NOTE: If you are ever interested in what open source software is being used by the ServiceNow platform just click on the link: Open Source Software. This contains all of the various open source titles AND their current versions. It makes interesting reading.

In looking for "hidden" information like this it is sometimes fruitful to go and see what applications are not active. Since we are interested in anything that might have the title cluster. First we see if there is perhaps something already available in the nav menu.

1. Type **cluster** in the nav filter.

© 2015 ServiceNow, Inc. All rights reserved.



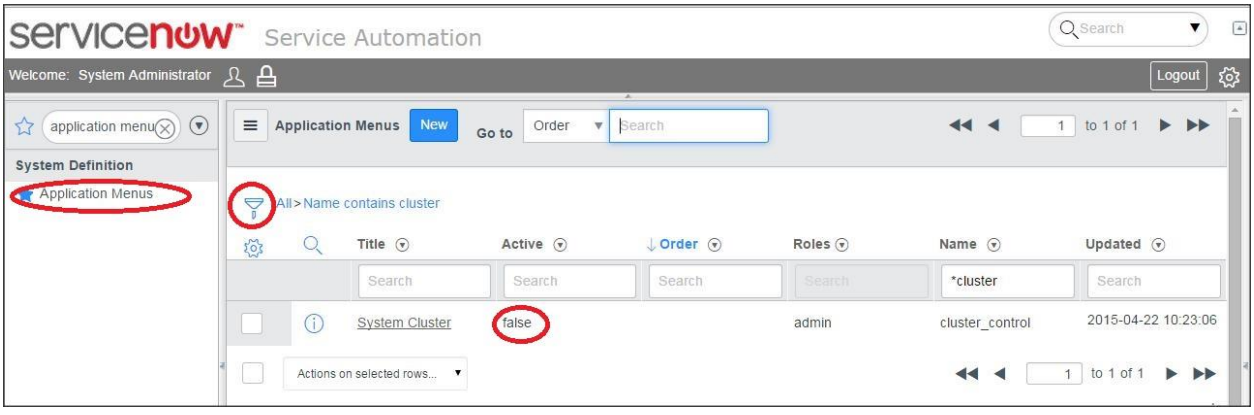
You will see that there are only entries under Configuration (CMDB), and MID Server. Neither of these are pertinent to our quest!

Next let's see if perhaps there is something we could use that just isn't turned on in the menu.

1. Type **module** in the nav filter.

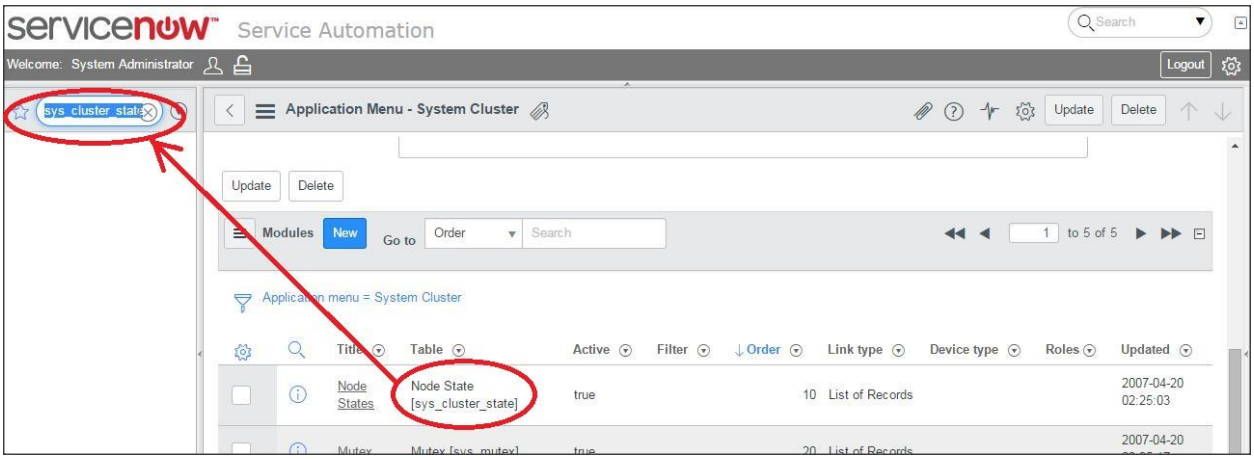
2. Navigate to **System Definition > Application Menus**

3. Modify the filter to search for everything with the word Cluster in the Name field. We notice that there is an application listed by the name of System Cluster. And it is not active! Looks promising.

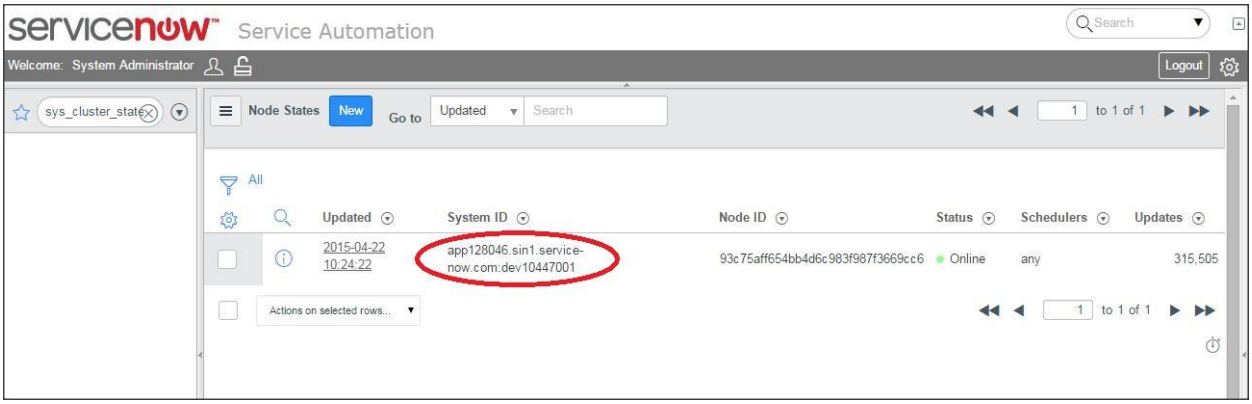


© 2015 ServiceNow, Inc. All rights reserved.

4. Click on **System Cluster**. You will note that there are five applications here. Since we know what is being displayed in stats.do is the current cluster node status, the first link Node States looks promising. It shows there is a table called: sys_cluster_state.

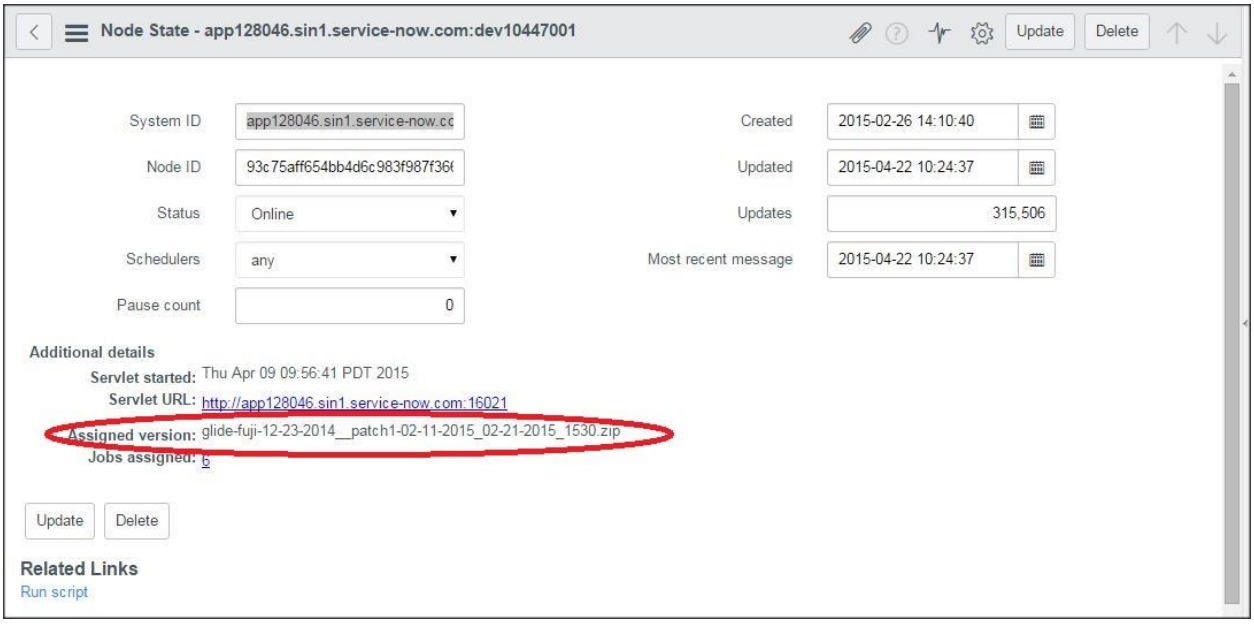


5. In the nav filter type in **sys_cluster_state.list**. The list view for that table will be displayed. Ah, better and better! Only one cluster node is displayed (the one we are logged into), and it has a system id that looks like the one we saw in the stats.do page.



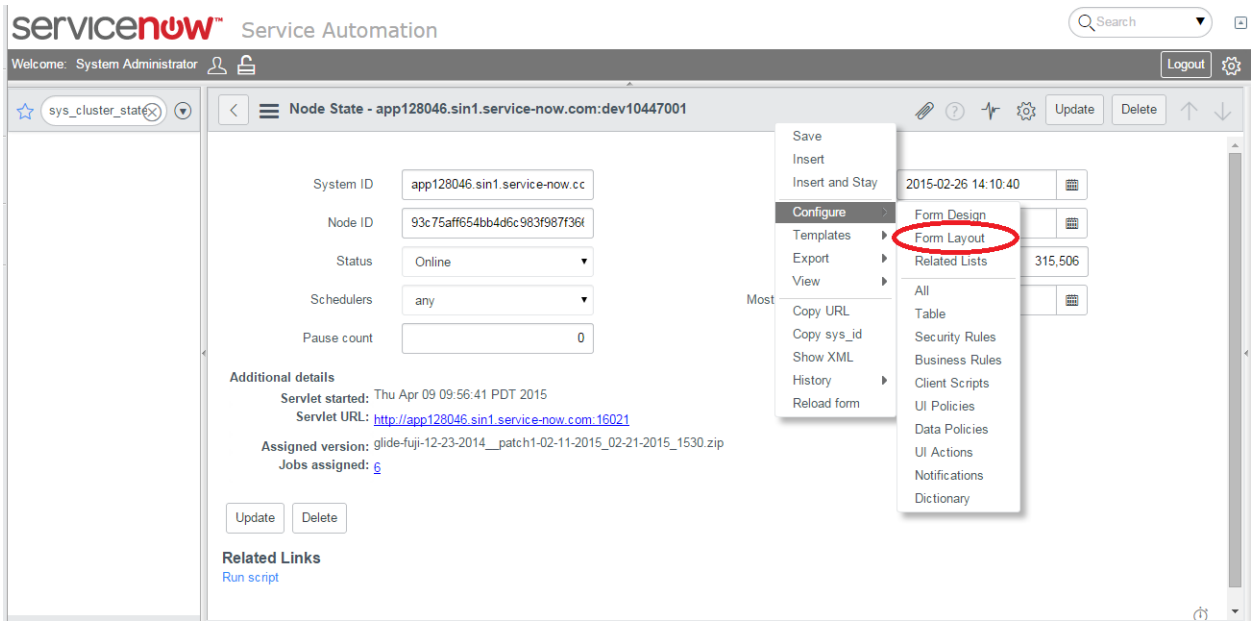
6. Click on the **Updated** field for that one record. The Node State form should appear. And this contains the patch info under the Assigned version field. Okay, we are cooking with gas now!

© 2015 ServiceNow, Inc. All rights reserved.



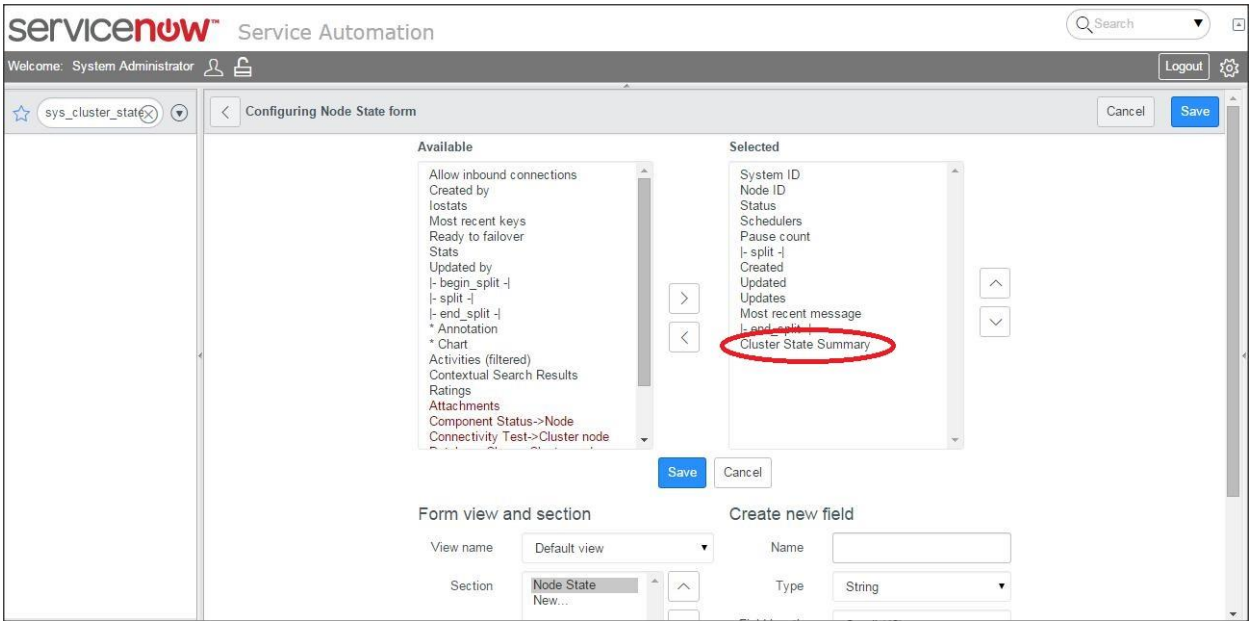
7. Right-click on the **Assigned version** field. You will find that there is NO field there! Right-clicking on Additional details does not show us a field either. So let's go find out what field is actually being referenced for this information to be displayed. Whenever you see something like this suspect it is actually a UI Macro that is being referenced and not a field.

8. Right-click on the **form header** to the display the context menu for the form.



9. Navigate to **Configure > Form Layout**. The form layout form will be displayed. Notice that there is an oddly named field: Cluster State Summary at the end of the "Selected" list. More-and-more this appears to be a UI Macro. To be sure we will check the dictionary for this table.

© 2015 ServiceNow, Inc. All rights reserved.



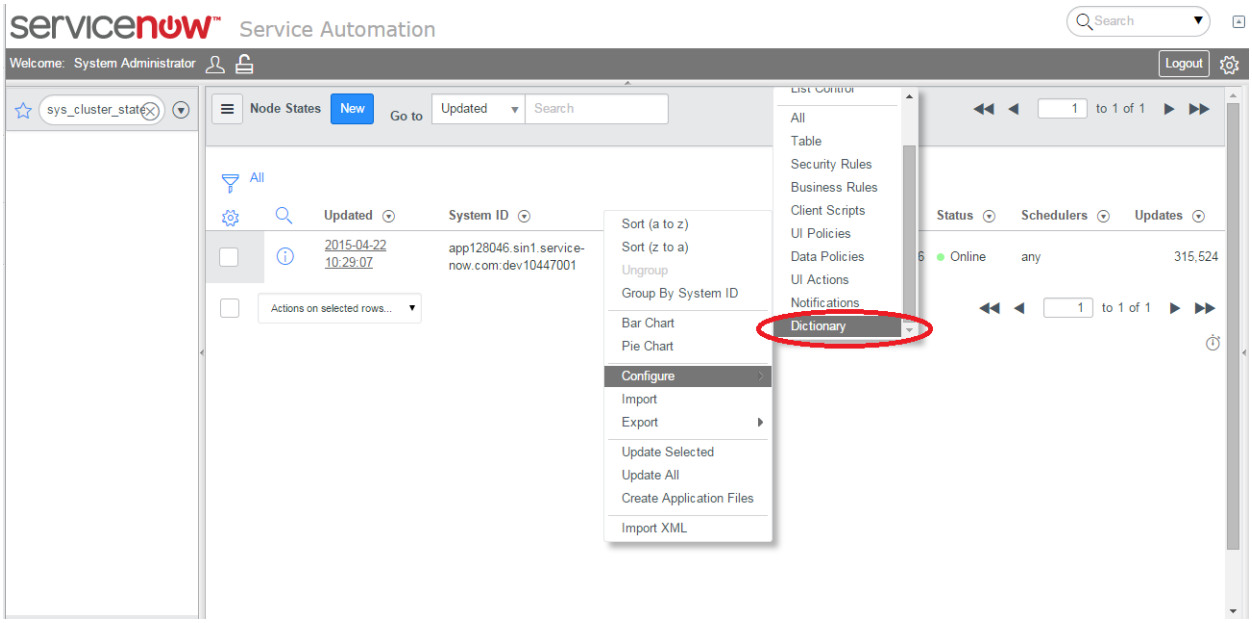
10. Click on the **Cancel** button.

11. Click on the **back-arrow** button on the form title bar to return to the sys_cluster_state list view.

12. Right-Click on the **list view header bar** to get the list view context menu.

13. Choose **Configure > Dictionary**. Look through the list for the field: Cluster State Summary. Not there. That pretty much clinches it. It has to be a UI Macro.

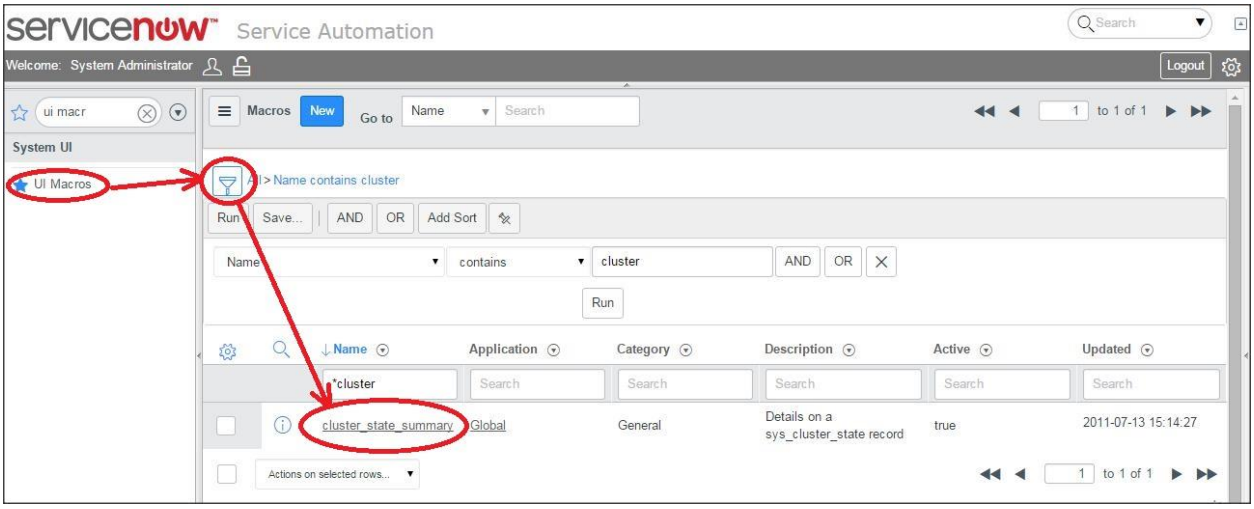
© 2015 ServiceNow, Inc. All rights reserved.



14. In the nav menu type **UI Macro**.

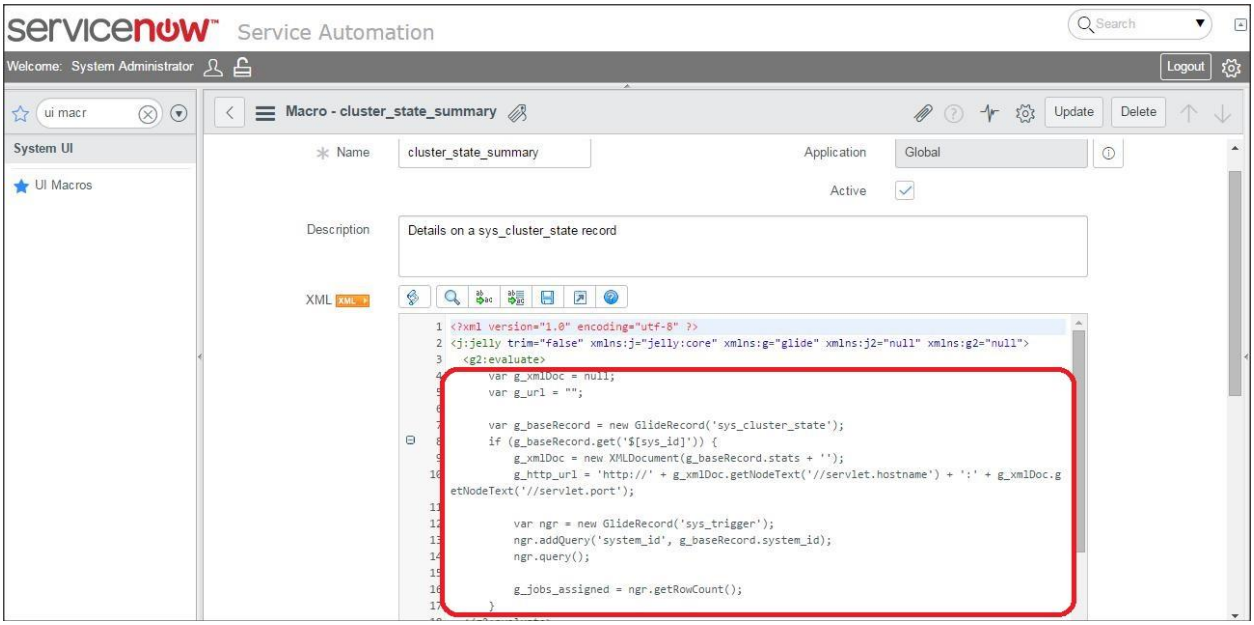
15. Navigate to **System UI > UI Macros**. The UI Macros list will appear.

16. Change the list filter to look for all names that contain the word: **cluster**. Bingo! There is one. cluster_state_summary!



17. Click on the **cluster_state_summary** entry. There is a section of code there that actually shows where things are being retrieved from. You will notice that this information is being kept in the sys_cluster_state table in a field named stats. It was there all along! The UI Macro just takes some of the data and formats it nicely for display. So it appears we may have our code example to retrieve the patch information we want!

© 2015 ServiceNow, Inc. All rights reserved.



18. Copy ALL of the code under the **<g2:evaluate>** tag (to the end of </g2:evaluate>). Just the code, and not the tags. Paste this in your favorite editor for safe-keeping (we will look at it now, and reference it as we construct our code).

ANALYSIS

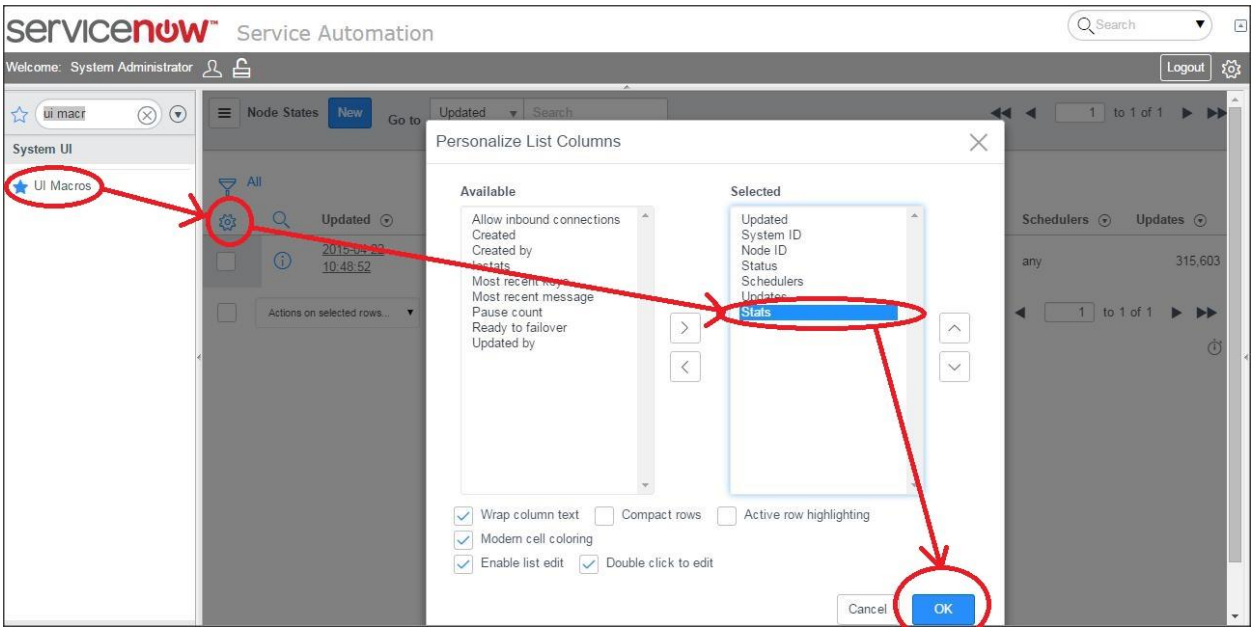
So it appears from this code snippet that the information we want is in the sys_cluster_state table, and is found in the stats field. Further since the XMLDocument object is used to pull the data from the field we can infer that the data there is XML. This STILL does not give us our patch info, but we are one step closer! Honest! Ok, back to the sys_cluster_state table!

19. In the nav filter type: **sys_cluster_state.list**. This will again display the list of cluster nodes.

20. Click on the **Gear** icon on the list view header bar. The list of available fields will be displayed.

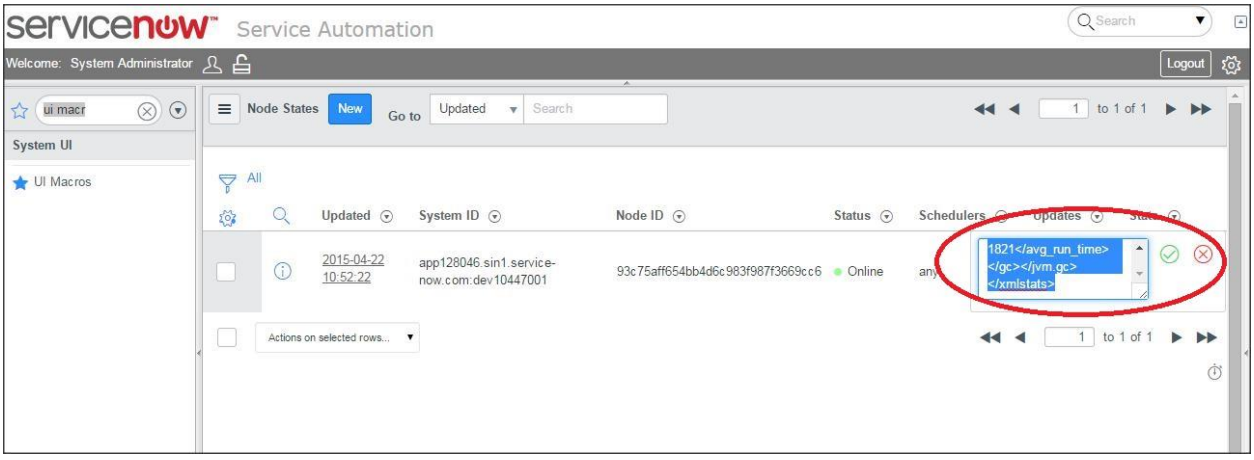
21. Click the **stats** field, and move it over into the displayed fields.

© 2015 ServiceNow, Inc. All rights reserved.



22. Click **OK**. The list view should now show the stats field.

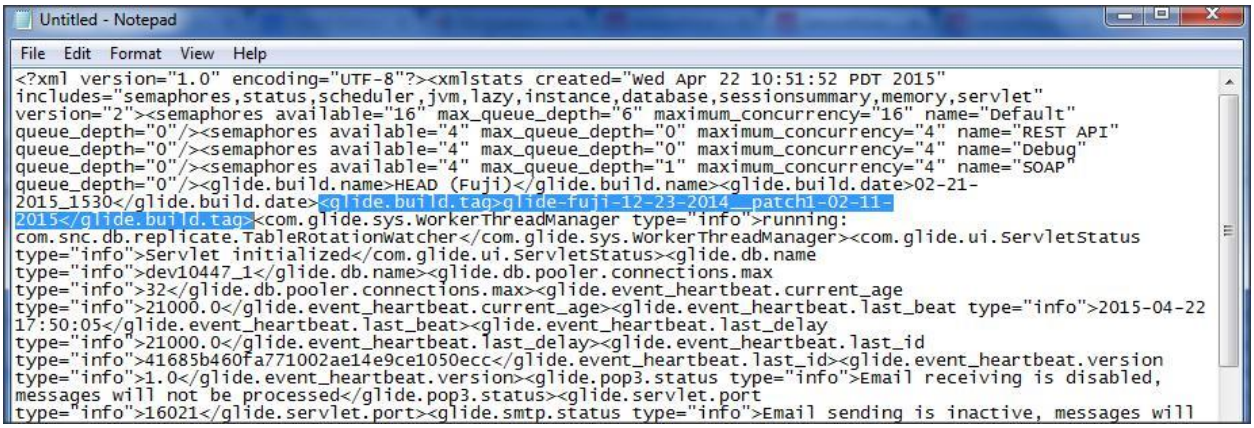
23. Double click on the **stats** field in the list view to edit the field. Copy the data there and past it into your favorite editor.



24. In the editor search for the word **patch**. We find it close to the beginning of the file, AND it appears to have our patch info!

25. Note the name of the XML field the "patch" data is contained in. It should be: **<glide.build.tag>.** This is the field we will retrieve our version and patch information from. We also see that the build date is right before it and is contained in: <glide.build.date>. We can grab that as well!

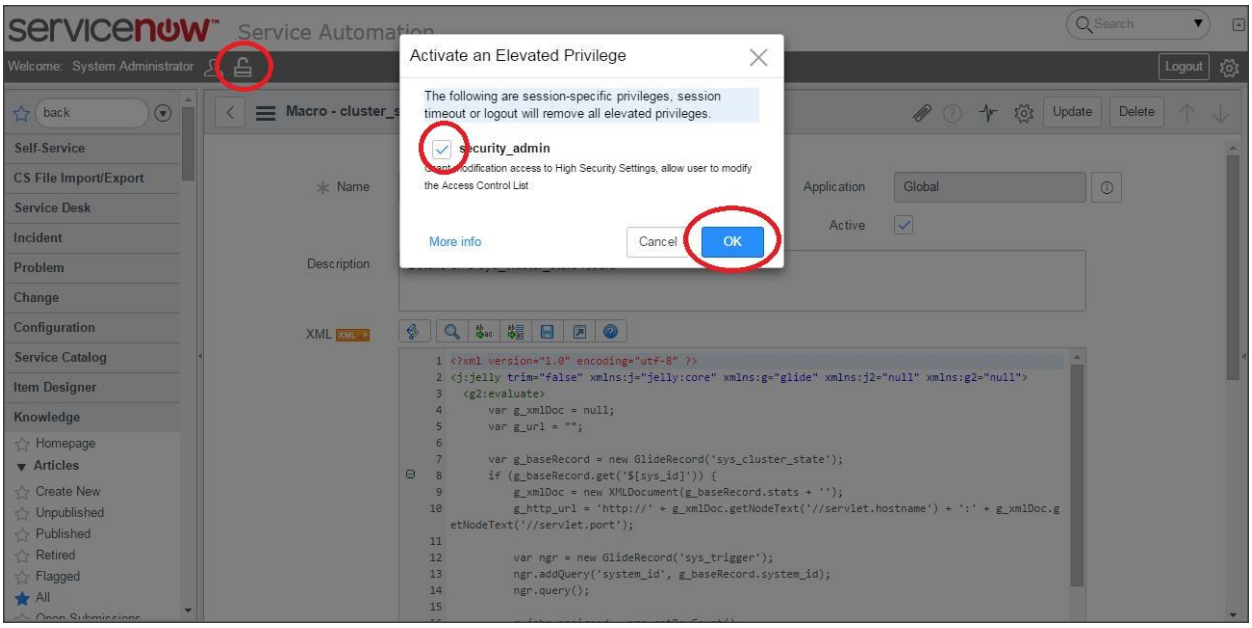
© 2015 ServiceNow, Inc. All rights reserved.



26. Go back to your browser.

27. Click on the **lock** icon on the banner bar (top left of the browser). The Activate an Elevated Privilege form will appear.

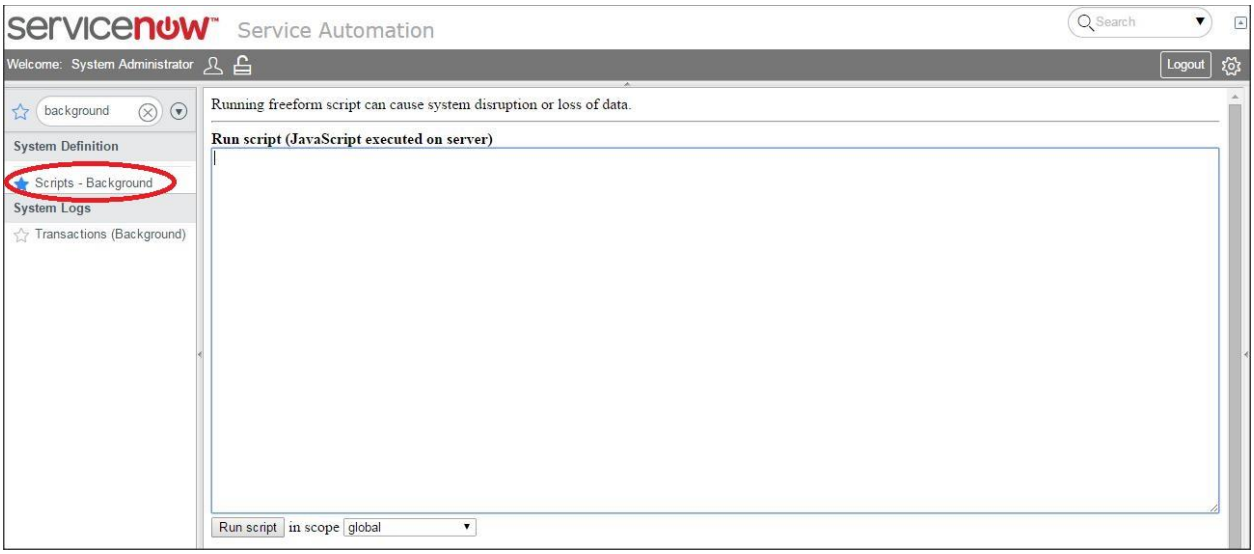
28. Click the **check-box** next to **security_admin**, and click the **OK** button. This will elevate your privileges, and allow you to see the Scripts - Background form.



29. In the nav filter type **Background**.

30. Click on the **System Definition > Scripts - Background** link. The Scripts - Background form will appear.

© 2015 ServiceNow, Inc. All rights reserved.



31. Enter the following code. Note the similarities to the code you copied into your notepad earlier:

```
var statsXML;
var clusterState = new GlideRecord('sys_cluster_state');
clusterState.setLimit(1);
clusterState.query();

if (clusterState.next()) {
   statsXML = new XMLDocument(clusterState.stats + '');
   versionAndPatch = statsXML.getNodeText('//glide.build.tag') + '';
   buildDate = statsXML.getNodeText('//glide.build.date') + ''
   gs.info('---> Version and Patch: {0}, date: {1}', versionAndPatch, buildDate);
}
```

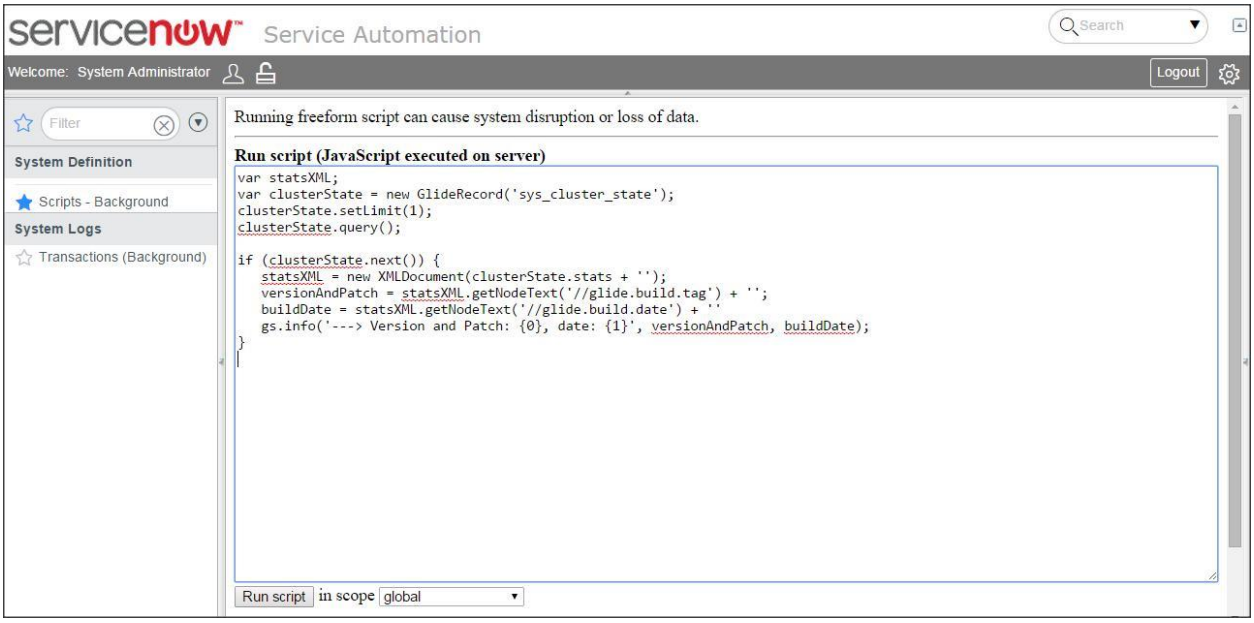
ANALYSIS

Since there is only one record we really don't have to use "setLimit" or "while". The setLimit is shown here as an example of how to limit the record set to the first record found. You will notice that there is an "if" instead of a "while" as we are only dealing with one record. This give two techniques to work with only the first record found. We load the XML field "stats" into an XMLDocument object in order to be able to more easily parse and retrieve the information we want. Notice how easy it is to now reference the version, patch and date information rather than searching one long XML string and parsing it ourselves. Use gs.info instead of gs.log as it is Application Scope safe. If you would like to read more then look up the following links:

Application Scoping
XMLDocument

© 2015 ServiceNow, Inc. All rights reserved.

Your code should look like this:



32. Click on the **Run Script** button at the bottom of the form.

33. This will "print" the Build Version, Patch and Date. We are there!



34. Close out (or save) your notepad session.

CHALLENGE

Add a new field on the Incident form that will contain the version and patch information (one field is all that is needed). Extend your Script Include to retrieve and pass back this information.

© 2015 ServiceNow, Inc. All rights reserved.